Day 31 of DevOps class

Agenda – Complete Learn about JSON:

1. JSON Means – JavaScript object notation

JSON is a syntax for storing and exchanging data.

JSON is Python

Python has a built-in package called json , which can be used to work JSON data.

Import the json module: import json

Convert from JSON to Python

Parse JSON is a text-based data format used to exchange and store data between web applications. It simplifies the data transmission process between different programming languages and platforms.

JSON string, you can parse it by using the json.loads() method

json.loads() function converts this nested json string into a nested Python dictionary. Is also know as Reading JSON from a file using Python.

Example:

import json  
  
names =('{"Name" : "John","Age" : 25,"place": "hyderabad"}')  
  
x = json.loads(names)  
  
print(x["Age"])

Convert from Python to JSON

If you have a python object, you can convert it into a JSON string by using the json.dump() method. Is also know as a writing JSON to a file in Python using json.dumps().

For writing json having two methods

Method 1:

The JSON package in Python has a function called json.dumps() that helps in converting a dictionary to a JSOn object. It takes two parameters:

* Dictionary – the name of a dictionary that should be converted to a JSON object.
* Indent – defines the number of units for indentation

Example:

import json

details = {  
 "Name": "John",  
 "Age" : 22,  
 "Gender" : "Male",  
 "Location" : "Hyderabad"  
}  
data = json.dumps(details)  
print(data)

Output:

{"Name": "John", "Age": 22, "Gender": "Male", "Location": "Hyderabad"}

Convert data types into JSON Strings:

* dict
* list
* tuple
* string
* int
* float
* True
* False
* None

Convert Python objects into JSON strings:

* dict – Dictionary to JSON String.

Example:

import json

details = {  
 "Name": "John",  
 "Age" : 22,  
 "Gender" : "Male",  
 "Location" : "Hyderabad"  
}  
data = json.dumps(details)  
print(data)

Output:

{"Name": "John", "Age": 22, "Gender": "Male", "Location": "Hyderabad"}

Method 2:

import json

details = {  
 "Name": "John",  
 "Age" : 22,  
 "Gender" : "Male",  
 "Location" : "Hyderabad"  
}

file\_path = 'js\_path'  
with open(file\_path, "w") as json\_file:  
 json.dump(details, json\_file)

* list – List to a JSON string

import json

name = ["sai", "john"]  
  
data = json.dumps(name)  
  
print(data)

Method 2:

import json

name = ["sai", "john"]

file\_path = 'js\_path'  
with open(file\_path, "w") as json\_file:  
 json.dump(name, json\_file

* tuple – Tuple to JSON string

import json

names = ("Apple", "Banana")  
  
data = json.dumps(names)  
  
print(data)

Method 2

import json

names = ("Apple", "Banana")  
  
file\_path = 'js\_path'  
with open(file\_path, "w") as json\_file:  
 json.dump(names, json\_file

When you convert from Python to JSON, Python objects are converted into the JSON (JavaScript) equivalent:

|  |  |
| --- | --- |
| **Python** | **JSON** |
| dict | Object |
| list | Array |
| tuple | Array |
| str | String |
| int | Number |
| float | Number |
| True | true |
| False | false |
| None | null |

Example:

import json

details = {  
 "Name": "John",  
 "Age" : 29,  
 "Gender" : "Male",  
 "Location" : "Hyderabad",  
 "Married" : True,  
 "Friends" : ("Max","Sam"),  
 "mileage": 22.5,  
 "Car models" : [  
 {"Car1" : "Ford", "Model" : "Mustang"},  
 {"Car2" : "Mini", "Model" : "clubman"},  
 ]  
}  
  
file\_path = 'js\_path'  
  
with open(file\_path, "w") as json\_file:  
 json.dump(details, json\_file)

Output:

{"Name": "John", "Age": 29, "Gender": "Male", "Location": "Hyderabad", "Married": true, "Friends": ["Max", "Sam"], "mileage": 22.5, "Car models": [{"Car1": "Ford", "Model": "Mustang"}, {"Car2": "Mini", "Model": "clubman"}]}

Format the Result:

details = {  
 "Name": "John",  
 "Age" : 29,  
 "Gender" : "Male",  
 "Location" : "Hyderabad",  
 "Married" : True,  
 "Friends" : ("Max","Sam"),  
 "mileage": 22.5,  
 "Car models" : [  
 {"Car1" : "Ford", "Model" : "Mustang"},  
 {"Car2" : "Mini", "Model" : "clubman"},  
 ]  
}

print(json.dumps(details, indent=3))

Output:

{

"Name": "John",

"Age": 29,

"Gender": "Male",

"Location": "Hyderabad",

"Married": true,

"Friends": [

"Max",

"Sam"

],

"mileage": 22.5,

"Car models": [

{

"Car1": "Ford",

"Model": "Mustang"

},

{

"Car2": "Mini",

"Model": "clubman"

}

]

}

when use the indent here notice the indentation between lines to visualize clearly

We can also define the separators, default value is (“, “, “:”), which means using a comma and a space to separate each object, and a colon and a space to separate keys from values:

Example:

import json

details = {  
 "Name": "John",  
 "Age" : 29,  
 "Gender" : "Male",  
 "Location" : "Hyderabad",  
 "Married" : True,  
 "Friends" : ("Max","Sam"),  
 "mileage": 22.5,  
 "Car models" : [  
 {"Car1" : "Ford", "Model" : "Mustang"},  
 {"Car2" : "Mini", "Model" : "clubman"},  
 ]  
}  
  
print(json.dumps(details, indent=3, separators=(".","=")))

Output:

{

"Name"="John".

"Age"=29.

"Gender"="Male".

"Location"="Hyderabad".

"Married"=true.

"Friends"=[

"Max".

"Sam"

].

"mileage"=22.5.

"Car models"=[

{

"Car1"="Ford".

"Model"="Mustang"

}.

{

"Car2"="Mini".

"Model"="clubman"

}

]

}

Order the result

The json.dumps() method has parameters to order the keys in the result:

Example:

import json

details = {  
 "Name": "John",  
 "Age" : 29,  
 "Gender" : "Male",  
 "Location" : "Hyderabad",  
 "Married" : True,  
 "Friends" : ("Max","Sam"),  
 "mileage": 22.5,  
 "Car models" : [  
 {"Car1" : "Ford", "Model" : "Mustang"},  
 {"Car2" : "Mini", "Model" : "clubman"},  
 ]  
}  
  
print(json.dumps(details, indent=3, sort\_keys=True))

Output:

{

"Age": 29,

"Car models": [

{

"Car1": "Ford",

"Model": "Mustang"

},

{

"Car2": "Mini",

"Model": "clubman"

}

],

"Friends": [

"Max",

"Sam"

],

"Gender": "Male",

"Location": "Hyderabad",

"Married": true,

"Name": "John",

"mileage": 22.5

}